Replica 3

Diogo Paulo Gomes Dias Leite Neves, up202108460, (40%), lógica por trás das dificuldades dos computadores, interface do menu e ajuda em várias funções da lógica base do jogo.

Rafael Costa Pires, up202208602, (60%), lógica base do jogo.

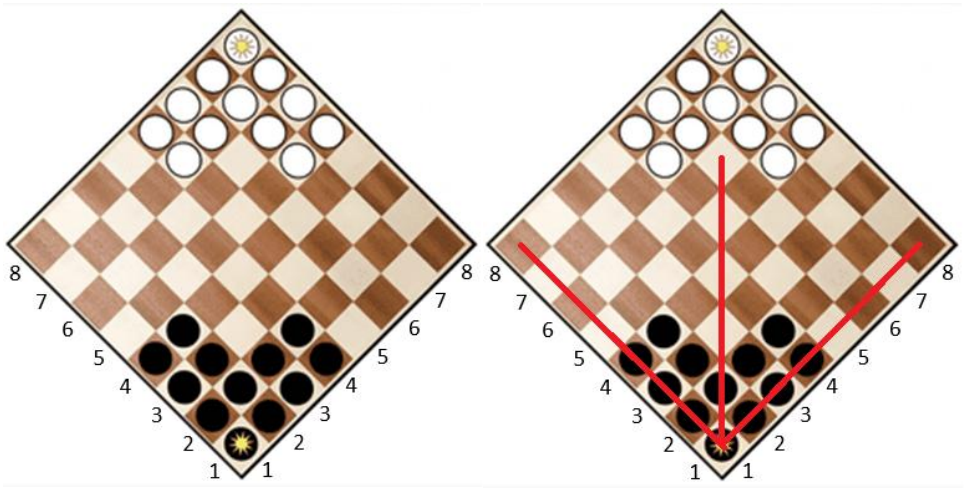
Instruções de instalação:

1) Dar unzip ao ficheiro;
2) Mudar a diretoria dentro do sicstus para a diretoria src
3) Compilar o código através do comando: [game].
4) Correr o comando: play.

Descrição do jogo

Estado inicial:

O jogo utiliza um tabuleiro 8x8 no qual cada jogador joga a partir de dois cantos opostos e cada jogador possui 11 peões e 1 rei no início do jogo como mostra a figura da esquerda:



Jogadas possíveis:

Cada jogador pode fazer 3 tipos de jogadas:

- Paço: Qualquer peça pode andar diretamente para a frente, ex.: (2,2),(3,3), para a frente e esquerda ex.: (4,2),(5,2), ou para a frente e direita ex.: (4,2),(4,3), se a casa em que o paço acabar for uma peça inimiga, essa peça é capturada e deixa de existir no jogo.
- Salto: Qualquer peça pode saltar por cima de peças aliadas para a frente ex.: (1,1),(3,3), para a frente e esquerda ex.: (1,1),(5,1) ou para a frente e direita ex.: (2,3),(2,5) se a casa em que o salto acabar for uma peça inimiga, essa peça é capturada e deixa de existir no jogo.
- Transformação: Os peões podem-se transformar em reis desde que tenham linha de visão para um rei aliado, apenas peças inimigas bloqueiam a visão como exemplificado na imagem da direita, um exemplo de uma peça que se pode transformar é a peça em (1,3).

Fim do jogo:

O jogo acaba quando um rei aliado se encontra no canto do inimigo ou quando um rei é capturado.

Considerações em relação a possíveis extensões ao jogo

O nosso código não é 100% modular mas para possíveis extensões do tamanho do tabuleiro seria apenas necessário mudar a função within_bounds(X,Y) que verifica se as coordenadas dadas estão entre 1 e 8.

Em relação à adição de novas regras seria também relativamente fácil, apenas seria necessário mudar a função validate_move() ao adicionar uma nova maneira de validar essa nova jogada, caso seja uma regra simples como, por exemplo, os reis podem-se mexer duas casas. É também possível criar regras diferentes entre as brancas e as pretas, de acordo com a estrutura do nosso código. Para regras mais complexas que envolvessem ações diferentes das que já existem, por exemplo, dois reis aliados podem combinar-se para formar uma peça mais forte que o rei, se um rei capturar o outro. Neste caso seria necessário alterar a função move() para que quando recebesse uma jogada que contenha nas posições inicial e final reis alterasse o tabuleiro da maneira correta.

Lógica do jogo

Representação interna e externa do jogo

No jogo, no estado em que está é representado da mesma maneira interna e externamente, Os espaços vazios são representados por '_', as peças pretas são representadas por 'b' e 'x' , peão e rei, respetivamente, e as brancas 'w' e 'y' peão e rei. No entanto, a aparência do jogo pode mudar muito facilmente ao alterar a função display_board() de modo a que quando receba uma matriz, represente os reis pretos, por exemplo, com o símbolo 'O'.

Representação de jogadas

Cada jogada é representada internamente assim: ((X1,Y1),(X2,Y2)), no qual X1,Y1 representam as coordenadas da peça que vai ser jogada e X2,Y2 as coordenadas de onde a peça vai ficar depois da jogada.

Interação com o utilizador

Depois do comando <play.> ser executado o utilizador depara-se com a seguinte interface:

```
##########################################

                REPLICA

1. Play Human vs Human
2. Play Human vs Computer
3. Play Computer vs Human
4. Play Computer vs Computer
5. Exit

##########################################
Choose an option (1-5):
|:
```

O utilizador escolhe uma opção ao escrever o número da opção seguida de um ponto final, por exemplo, "1." esta opção leva diretamente a um jogo entre dois jogadores. Se o jogador escolher uma opção que inclui computadores é pedido outro input para escolher se os computadores estão na dificuldade 1 ou 2:

```
Starting Human vs Computer mode...
Choose Computer Level:
1. Level 1 (Random)
2. Level 2 (Greedy)
|:
```

Depois de escolhida uma opção que inclui pelo menos um jogador, quando é o seu turno, o jogador pode inserir as suas jogadas em dois formatos diferentes ((X1,Y1),(X2,Y2)) ou ((X1,Y1),X2,Y2). X1 e Y1 representam as coordenadas de peça que vai ser jogada e X2 e Y2 as coordenadas de destino para a peça:

```
Choose an option (1-5):
|: 1.
Starting Human vs Human mode...
Game starting... (Player 1: human, Player 2: human)
White to play
8 _ _ _ _ w w w y
7 _ _ _ _ w w w w
6 _ _ _ _ _ _ w w
5 _ _ _ _ _ _ w w
4 b b _ _ _ _ _ _
3 b b _ _ _ _ _ _
2 b b b b _ _ _ _
1 x b b b _ _ _ _
  1 2 3 4 5 6 7 8
Number of valid moves: 43

Insert your play:
Enter your move in the form ((X1,Y1),(X2,Y2)):
|:
```

Quando um jogo acaba, o menu principal aparece de novo para que o utilizador posso mudar de modo de jogo se quiser.

Conclusão

Ao acabar o projeto concluímos que ficámos contentes com a implementação do jogo e com a maneira como é apresentado ao jogador. A maneira como os computadores interagem com o tabuleiro e com o jogador é satisfatório, mas a implementação do segundo nível do computador (greedy) não é ótima. No que toca ao input do jogador, notámos também que quando um jogador tenta jogar uma peça adversária o input não é corretamente filtrado e o programa falha, algo que não tivemos tempo para corrigir. No futuro poderíamos mudar a estrutura de certas funções para que a adição de novas regras seja ainda mais fácil, por mais que neste momento seja acessível, sentimos que é possível tornar a adição de novas regras ainda mais fácil.

Bibliografia

https://boardgamegeek.com/boardgame/427267/replica